Miguel López Manzanares
Enric Ferrera González
Yassin Nakmouche Sahli M'Ghaiti

UAB The Hack: FGC

Planteamiento

En esta hackathon desde los Ferrocarriles de la Generalitat se nos planteaba un problema abierto, posibilitar mediante el uso de las TIC una movilidad más inclusiva.

Fases de trabajo:

1. BrainStorming
2. Problema a solucionar
3. Diseño del software
4. Desarrollo del software
5. Paso a dispositivo móvil
6. Conclusiones, aprendizajes y mejoras

Fase 1 BrainStorming:

Consideramos esta fase la más importante, más incluso que la implementación técnica, ya que sin una buena idea que plantee un problema real, el resto carece de sentido.

Le hemos dedicado una gran parte del tiempo a pensar y perfeccionar la idea final, pero antes de esas hemos barajado muchas otras, las cuales hemos tenido que descartar tras ver su poca utilidad o, sorprendentemente, ya estaban implementadas. A continuación dejamos un breve recopilatorio de las ideas más interesantes y motivos de descarte:

- IA que te ayuda a comprar el billete, el botón de información da completo, ya existe un servicio de atención al cliente que puede tomar control del terminal y comprar por ti.
- Sensores en los asientos para detectar cómo de ocupado está el tren, irrelevante, ya tenemos esta métrica
- Visión por computador para detectar cómo de ocupado está el tren, irrelevante, ya tenemos esta métrica

Miguel López Manzanares
Enric Ferrera González
Yassin Nakmouche Sahli M'Ghaiti

Fase 2: Problema a solucionar:

Hemos detectado que la gente sorda o sordomuda, al presionar el botón ayuda, tienen que esperar a que una persona física venga a ayudarlos. Esto se realiza mediante el color del botón de ayuda, dependiendo de este podemos saber el estado de la llamada. Queremos evitar esta complicación creando una aplicación de Speech-To-Text(STT) y Text-To-Speech(TTS) con la cual se puedan comunicar con el operador utilizando el teléfono móvil.

Esta idea viene reciclada de una anterior, el mismo concepto, pero conectando, mediante un chip NFC al lado del botón de información, a una página web con un chat en vivo con atención al cliente. La convertimos en la actual al detectar que podíamos solucionar un problema más grande. El problema que vamos a solucionar aplica a todos los interfonos que uno pueda encontrar.

Fase 3 Diseño del software:

El problema plantea un reto muy nuevo para nosotros, nunca hemos trabajado con TTS ni STT y nunca habíamos intentado pasar el código ejecutado en el terminal a una app.
Tras mucha investigación hemos elegido las siguientes librerías para solucionar el problema:

- SpeechRecognition
- gTTS (google Text-To-Speech)
- pydub
- playsound
- kivy: utilizado para la interfaz y para convertir en app

Miguel López Manzanares
Enric Ferrera González
Yassin Nakmouche Sahli M'Ghaiti

Fase 4: Desarrollo del software:

Hemos desarrollado todo el software de la parte de Back-End con Python como por ejemplo tomar la entrada de voz y convertirla a texto y viceversa.
Aquí la función para ejecutar el STT:

```python
def speech_to_text(self, instance):
    self.label.text = "Por favor, habla ahora..."
    threading.Thread(target=self._perform_speech_to_text).start()

1 usage
def _perform_speech_to_text(self):
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        audio = recognizer.listen(source)

    try:
        text = recognizer.recognize_google(audio, language="es-ES")
        Clock.schedule_once(lambda dt: self.update_label(f"Texto reconocido: {text}"), timeout: 0)
    except sr.UnknownValueError:
        Clock.schedule_once(lambda dt: self.update_label("No se pudo entender el audio"), timeout: 0)
    except sr.RequestError:
        Clock.schedule_once(lambda dt: self.update_label("Error con el servicio de reconocimiento"), timeout: 0)
```

De estas 2 funciones la línea más importante es la que está marcada en rojo en la imagen superior, porque "recognize" es una instancia de la librería de google "speech_recognition" y la "recognize_google" que es el método de Recognize que se encarga de enviar el audio grabado a los servidores de Google para su procesamiento y recibe el texto correspondiente.

Aquí la función para ejecutar el TTS:

```python
def text_to_speech(self, instance):
    text = self.text_input.text
    if text:
        threading.Thread(target=self._perform_text_to_speech, args=(text,)).start()

1 usage
def _perform_text_to_speech(self, text):
    tts = gTTS(text=text, lang='es')
    filename = "temp_audio.mp3"
    tts.save(filename)
    audio = AudioSegment.from_mp3(filename)
    play(audio)
    os.remove(filename)
    Clock.schedule_once(lambda dt: self.update_label("Reproducción completada"), timeout: 0)
```

Aquí la línea más importante es la que se marca con la flecha en rojo la cual gTTS es una clase de la librería gTTS que se utiliza para convertir texto a audio.

Por la parte de Front-End para darle un poco de forma y color a nuestro programa de STT y TTS hemos usado kivy,que es una biblioteca de Python que permite desarrollar aplicaciones

multi-touch para una amplia gama de dispositivos, incluyendo sistemas operativos como Windows, OS X, Linux, Android e iOS.

Aquí dejamos parte del código de la interfaz gráfica y el resultado final de la interfaz gráfica.

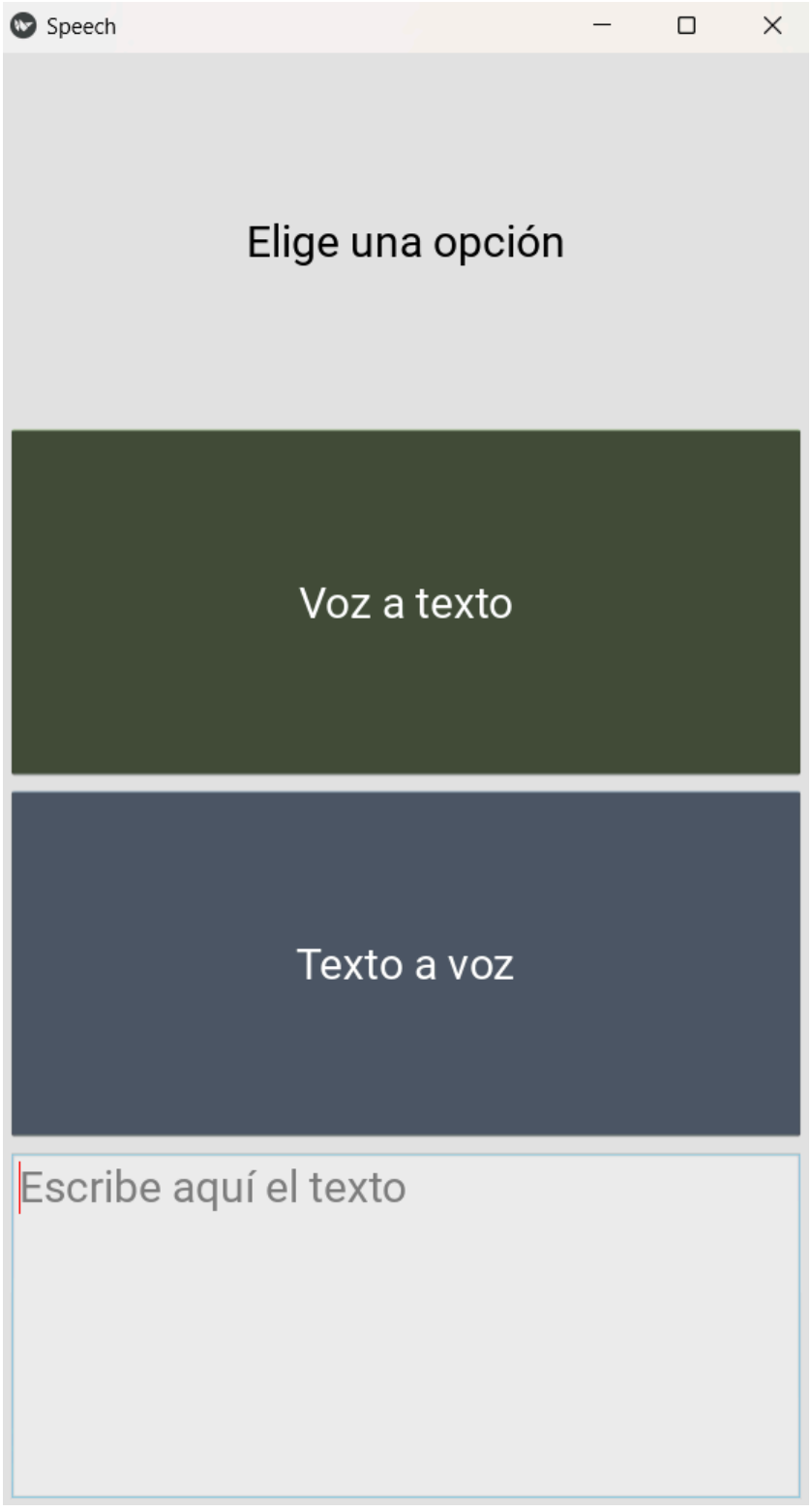
```
class MainScreen(Screen):
    def __init__(self, **kwargs):
        super(MainScreen, self).__init__(**kwargs)
        layout = BoxLayout(orientation='vertical', padding=10, spacing=10)

        with layout.canvas.before:
            Color( *args: 1, 1, 1, 0.9)
            self.rect = Rectangle(size=(1920,1080), pos=layout.pos)

        self.label = Label(text="Elige una opción",
                           font_size='24sp',
                           color=[0, 0, 0, 1])
        layout.add_widget(self.label)

        btn_speech_to_text = Button(
            text="Voz a texto",
            font_size='24sp',
            background_color=[0.741, 0.894, 0.655, 1],
            color=[1, 1, 1, 1]
```

Miguel López Manzanares
Enric Ferrera González
Yassin Nakmouche Sahli M'Ghaiti



Miguel López Manzanares
Enric Ferrera González
Yassin Nakmouche Sahli M'Ghaiti

Fase 5: Paso a dispositivo móvil

En un principio, investigando, habíamos descubierto kivy, una librería de python que permite desarrollar app en este lenguaje. Ha sido muy útil para el front-end, pero nos ha creado muchos problemas, los cuales no hemos sido capaces de solucionar a la hora de lanzar esta aplicación en móvil, pero kivy nos ha sido útil para crear un prototipo el cual sirve para tener una idea de cómo será la App, hemos buscado otras opciones, la que hemos estado desarrollando es una web con gradio, pero al tener la plataforma los sockets cerrados no hemos sido capaces de hacer una live demo en el teléfono móvil.

Fase 6:

1. Aprendizajes mejoras y conclusiones

Tras acabar el proyecto, podemos ver las diferentes áreas y conocimientos que hemos tenido que adquirir. Previo a la versión STT-TTS estuvimos creando una web de chat (JSON, html, javascript, sockets) en tiempo real e investigamos sobre NFC's. Una vez cambiado el objetivo del proyecto, hemos aprendido pinceladas del STT, TTS, front-end y desarrollo móvil con python(kivy).

2. Mejoras

- Una vez hecho el prototipo, detectamos diferentes áreas de mejora.
- Detección y traducción del lenguaje
- Correcta implementación en una app

3. Conclusiones

No ha sido nada fácil, ya que no teníamos apenas conocimientos sobre las tecnologías que queríamos tratar y estamos satisfechos de haber podido sacar un prototipo funcional. Por otro lado, estamos muy frustrados por nuestra incapacidad para que el programa se ejecute en una app/apk, hemos tenido muchos errores sobre un proceso que parecía sencillo a priori. Hemos barajado y probado otras opciones cómo una página web, pero la falta de conocimientos nos ha limitado mucho. Valoramos la experiencia muy positivamente, hemos tenido que forzarnos a aprender cosas muy lejos de nuestras zonas de confort, conocer mucha gente nueva y pasar un buen rato con nuestros compañeros de equipo.